iRobot Education's Create 3

Web Playground for Python

Explore the following code snippets used to program the Create 3 Education Robot in the Web Playground for Python.

`@event(robot.when_bumped, [True, False])` `async def when_bumper(robot):`	Robot senses input from the bumper sensors on the left and right side. [True, False]: Left Side of Bumper Pressed [Fasle, True]: Right Side of Bumper Pressed []: Any Bumper Pressed
`@event(robot.when_touched, [True, False]` `async def when_touched(robot):`	Robot senses input from the bumper sensors on the left and right side. [True, False]: Button 1 Pressed [False, True]: Button 2 Pressed
`@event(robot.when_play)` `async def when_play(robot):`	Commands run when Play button is pressed.
`await robot.move(16)`	Robot moves forward specified number of centimeters. Example shows 16cm.
`await robot.arc(Robot.DIR_LEFT, 90, 4)` `await robot.arc(Robot.DIR_RIGHT, 90, 4)`	Robot drives forward or backward in an arc, either clockwise (RIGHT) or counter-clockwise (LEFT) for a specified number of degrees along a curve of a specified radius. Example #1 shows robot driving counter-clockwise 90• around a curve with a 4cm radius.
`await robot.turn_left(90)` `await robot.turn_right(90)`	Robot rotates clockwise (RIGHT) or counter-clockwise (LEFT) for a specified number of degrees. Example #1 shows robot driving counter-clockwise 90•
`await robot.navigate_to(16 , 16)`	Robot navigates across an invisible grid of centimeter increments. The origin of (0, 0) is automatically set to the robot's starting position, with the x-axis aligning to the robot's center button. Example shows robot driving to the coordinate (16, 16)
`await robot.get_position()`	Robot identifies its current location based on an invisible grid of centimeter increments. The x-axis aligns to the robot's center button. Example will return robot's current XY coordinates.

`await robot.reset_navigation()`	Command reset robot's navigation grid origin of (0, 0) to current position of robot, with the x-axis aligning to the robot's center button.
`await robot.set_lights_rgb(0, 0, 255)` *await robot.set_lights(Robot.LIGHT_ON, Color(R, G, B))* *await robot.set_lights(Robot.LIGHT_SPIN, Color(R, G, B))* *await robot. set_lights(Robot.LIGHT_BLINK, Color(R, G, B))* *await robot. set_lights(Robot.LIGHT_OFF, Color(R, G, B))*	Command controls the light pattern (on, spinning, blinking or off) and RGB color of the LED's in the robot's Ring Light. *Example #1 shows a convenience function with light pattern automatically set to "on," as this is the most commonly used light status. The color of the lights is pure blue.
`await robot.set_wheel_speeds(10, -10)`	Command sets the speed (in cm/s) and direction of robot's wheels on either side. Example shows robot's left wheel (on the same side as Button 1) driving forward 10 cm/s and right weel (on the same side as Button 2) driving backward 10 cm/s. The robot will rotate in place, clockwise.
`await robot.wait(0.5)`	Command specifies a delay in seconds before moving onto the next line. Example tells robot to wait .5 seconds before reading next line.
`await robot.play_note(440, 0.25)`	Robot will play a musical note of a specified frequency for a specified duration. Example plays the note A4 for .25 seconds.
`await robot.stop_sound()`	Robot will stop all sounds currently playing.